AN OVERLAPING COMMAND COMMITTING METHOD OF DYNAMIC CYCLE PIPELINE

FIELD OF THE INVENTION

The present invention relates to the design of various chips in the technical fields such as network communications, in particular, to a command committing method of dynamic cycle pipeline processing for chips.

BACKGROUND OF THE INVENTION

In the design of communication chips, in order to meet the needs of the rapid increase of the speed and capacity of communication networks, the pipeline structure processing has been adopted in more and more chips. The pipeline technology is a technology of having the parallelism with space and time in computer technology, which breaks down one sequence process into several sub-processing steps, each of which can effectively perform simultaneously on the special independent module. These sub-processing steps are called stages, while every pipeline stage is composed of current register and hardware processing module (pure logic circuit), the former providing input to the latter, and the output of the latter point to the current register of the next stage; under the effect of clock pulse, every stage transmit the result from the completed processing to the next stage simultaneously. Committing the command to the pipeline promptly, so as to assure that the command is continuously flowing in the pipeline, is the key of obtaining highly effective pipeline.

Figure 1 illustrates the command committing method of ring-form pipeline in the prior art, the ordinate stands for the stage of the pipeline, including six stages of pipelines, while the abscissa stands for time, A, B, C, D, E, F and G stand for the commands performed. The inserting position of the command is the first stage, and the command exiting position is the third stage of the pipeline. In case that all the pipeline stages are full, A, B, C, D, E and F are all performing in the pipeline, each of the current inset processing units inserts new command only after the pipeline commands entirely exit the cycle pipeline, as is illustrated by the figure, the new Command G is inserted after Command A completely exits the cycle pipeline, the "bubble" appears from the fourth stage to the sixth stage of the pipeline. Therefore, it cannot be guaranteed that the pipeline is flowing continuously,

which thus affects the utilization efficiency of each elements of the pipeline and the parallelism of the command performance.

SUMMARY OF THE INVENTION

The objective of the present invention is to provide an overlapping command committing method of dynamic cycle pipeline, which can avoid the appearance of bubble, and thus can improve the parallelism of the command performance and the utilization efficiency of every part of the pipeline.

In order to accomplish the above objective, the present invention provides an overlapping command committing method of dynamic cycle pipeline for a chip having pipeline structure, comprising the following steps:

(a) reading the command from command buffer;

(b) decoding the command, judging whether it is a illegal command or not, if yes, then returning to step (a); otherwise, performing the next step;

(c) preprocessing the operator of the command, preparing the initial operator of each pipeline, and storing it into the initialization register;

(d) judging whether the pipeline is not full, if it is not full, directly inserting new command and then end; otherwise, waiting for the pipeline command exiting signal from the last pipeline period before exiting;

(e) after receiving the exiting signal, judging whether there is command relevance between the new command to be inserted and the old command to exit, if yes, then inserting the new command after the said old command exits, and then end; otherwise, performing the next step;

(f) in the last cycle of the pipeline by the said old command, committing the new command to the pipeline.

The above method can be characterized as: the aforesaid command relevance means that the new command and the old command cannot share the hardware processing module of one pipeline stage.

The above method can be characterized as: in the aforesaid step (e), it also judges in which stage of the pipeline stage the new and old commands should perform field switch, and finishes the field switch in the corresponding pipeline stage where the new and old commands overlap.

In order to resolve the conflict on field, the above method can be characterized as: in the aforesaid step (e), it also judges whether there is field conflict between the new command and the old command, if the field conflict exists, then adds the field of the new command into the pipeline when committing, while the field of the old command enters into field branch and maintains until the last time the old command use this field; if there is no field conflict, then finishes the field switch on the corresponding pipeline stage after committing.

In order to better perform the above method, the present invention also provides a communication chip with the cycle pipeline structure, comprising: interface of host computer, input buffer, command processing unit, result unit; the said command processing unit comprises: command interpreter and pipeline performing unit; wherein, the said command interpreter further comprises: command buffer controllers which are connected in order, command register, operator processing unit, pipeline initialization register and control automaton; the said control automaton controls the said command buffer controller to read a command from the command buffer, and stores the said command into the command register; the said control automaton decodes the said command, and controls the processing unit of operator to prepare initial operator of each pipeline according to the type of said command, and stores it into the said pipeline initialization register.

Therefore, by using the overlapping command committing method of dynamic cycle pipeline, the interval between each committing of command is reduced, so as to improve the parallelism of pipeline performing unit and reduce the processing period of command in the chip, and thus let the chip process more command in unit time. Furthermore, the command committing elements and the pipeline performing elements can each perform continuously, so that the time of the idle status of elements has been reduced to the minimum.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates the command committing method of the pipeline in the prior art;

FIG. 2 is a block diagram of a chip on which embodiments of the invention may be implemented;

FIG. 3 is a block diagram of the command interpreter in FIG.2;

FIG. 4 is a flow chart of the overlapping command committing of the command interpreter in an embodiment of the invention;

FIG. 5 is a schematic diagram of the overlapping command committing in an embodiment of the invention;

FIG. 6 is a schematic diagram of the hardware structure for field switching in the embodiments of the invention;

FIG. 7 is a schematic diagram of the hardware structure for field branching in the embodiments of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Hereunder the command committing of pipeline in a communication chip is shown as an example to describe the implementation of this invention in detail. The structure of the chip is shown in Fig. 2, wherein the host computer interface 11 in this chip 1 receives various commands such as: reading/writing access of the memories, the reading/writing access of the control register, the search command of different calculation methods, etc from the host computer, and stores them into the input buffer 12, and the command processing unit 13 performs the operation of these commands; once the performance of the commands is completed, the performance results of the commands are stored in the result_buffer 14, to be retrieved by the host computer.

The chip in this embodiment supports the following commands:
Reading/writing memories command;
Reading/writing control register command; and
Search command, which support 8 kinds of calculation methods.

The command processing unit 13 is composed of two parts: command interpreter 131 and the pipeline performing element 132. The command interpreter 131 serves to draw the command word and operator from the command buffer and data buffer of the input buffer 12, and to send them into the pipeline.

Within the pipeline, the operation sequence of the performance of each command is fixed, but the time when the command enters into the pipeline for performance is determined by the pipeline status. The switch between different command operations on the pipelines is continuous, however, due to the possible cycle operation, the sequence of commands entering the pipeline may be inconsistent with the sequence of commands exiting the pipeline after the completion of performance. After the completion of the command performance, all the command results are retrieved in order by the result_buffer 14. The sequential operation of each command requires the continuous performance of

several pipeline periods, whose cycle course is also controlled by the pipeline. The pipeline in this embodiment is divided into six stages, which can contain six commands at the same time, and the field of each command is flowing along the stage in the pipeline.

The command interpreter 131 and the pipeline performing unit 132 are working in parallelism: while the pipeline performing unit 132 is performing various command operations in parallelism, the command interpreter 131 is drawing the commands from the command buffer, conducting preprocessing, and commit one irrelevant command at one time to the pipeline by the overlapping committing method.

The so called overlapping committing method means inserting a new command in the last cycle of certain command before exiting the pipeline, as is shown in Fig. 5, Command G is inserted in the last cycle of Command A before exiting, so as to avoid the appearance of bubble. In order to carry out the above method, it is required to provide a command exiting signal as early as possible during the pipeline period before the last cycle before the command exiting the pipeline; and it shall ascertain that there is no command relevance between the newly inserted command and the exiting command; in addition, between the inserted and exiting commands, the possible conflict of command fields formed by registers (referring to the current registers) shall be resolved.

The constituting structure of the command interpreter 131 is shown in Fig. 3, including the command buffer reading controller 21, the command register 22, the operator processing unit 23, the pipeline initialization register 24 and the control automaton 25; the following is an explanation of the progress the command committing method of the command interpreter 131 in this embodiment with reference to this structure, as is shown in Fig. 4, the method according to this embodiment comprises the following steps:

Step 100, the control automaton 25 of the command interpreter 131 controls the command buffer reading controller 21 to read commands from the command buffer, and stores them into the command register 22;

Step 110, the control automation 25 decodes the command, delete the illegal command which has incorrect command code or carries unreasonable command parameters;

Step 120, the control automaton 25 controls the operator processing unit 23 to prepare the initial operator of each pipeline based upon the type of command, and stores it into pipeline initialization register 24;

Step 130, judging whether the pipeline is not full (i.e., less than six commands), if the pipeline is not full, then conducting Step 140, otherwise (i.e., containing six commands), conducting Step 150;

Step 140, directly inserting the command, and then end;

Step 150, waiting for the command exiting signal provided by the pipeline;

Step 160, once receiving the command exiting signal, judging whether there is command relevance between the new command to be inserted and the old command to exit, if there is, then conducting Step 170, otherwise, conducting Step 180;

Step 170, inserting the new command after the old command exits, and then end;

Step 180, inserting the new command to the pipeline during the last cycle of the old command before exiting.

In case the initial operator in Step 120 is a reading/writing memories command, it is required to provide the memories access address and necessary written data; in case the initial operator in Step 120 is a search command, it is required to provide the key word for searching and the adopted calculating method for searching; in case the initial operator in Step 120 is a reading/writing control register command, it is required to provide the register access address and necessary written data. Besides, it is also required to provide the initial status when entering the pipeline for all kinds of commands, that is, to map the command word to the current register value of the pipeline when the command enters the pipeline.

In Step 150, in order to ascertain that the command interpreter promptly determines the time of inserting the command, it is required to provide a command exiting signal as early as possible during the pipeline period before the last cycle of the command exiting the pipeline, to help the command interpreter to determine the time of inserting the command. The position of releasing the pipeline exiting signal is fixed, which in this embodiment is at the first two stages when the new command enters the pipeline stage. Taking Fig. 5 as an example, the new Command G is inserted at No. 1 pipeline stage, thus the pipeline exiting signal of the old Command A is released at the fifth stage of the one pipeline period before, and when the Command A to exit flows to the sixth stage pipeline, the command is added with judgment logic to determine whether or not to insert the new Command G.

In Step 160, the non-existence of command relevance means that the command to exit and the command to be inserted shall not access the same element, for example, the inserted command and the exiting command shall not use the hardware processing module element

of the same pipeline stage at the same time, which can be predetermined by the operation flow of commands.

In Step 180, when the new command is inserted, since the field of the inserted command is sequentially added into several pipeline stages for operations, while the field of the exiting command may also be in use at the last several stages, it thus may cause conflict that the two may have to use the same one command current register at the same time. It is determined by the operation flow of various commands which fields shall be used in which pipelines, and the conflict between the new and old fields can be predetermined based on the overlapping command sequence.

In most applications, the same field may just be switched to the field of new command before the possible use of the inserted command and after the last use of the exiting command. The hardware structure of such field switching is shown in Fig. 6. The current register 61 is connected to the multi-route switch 63 through the processing module 62, while the new current register (the new command) 64 is directly connected to the multi-route switch 63, and the input of the multi-route switch 63 is connected to the current register 65 of the next stage of pipeline. Normally, the content of the current register 61 of the last stage of pipeline subject to the processing of the processing module 62 and then enters into the current register 65 of the next stage of pipeline, while at the time of command overlapping, the new current register 64 enters into the current register 65 of the next stage of pipeline, and the field of the old command (located at the current register of the last pipeline) is deserted. In Fig. 5, all the three stages of command overlapping pipelines (pipeline stages 1, 2 and 3) can generate the aforesaid switch of current registers, and it can be determined in which stage the switch shall happen according to the specific operation command in the stage before inserting a new command.

In some applications, the field conflict is inevitable, i.e. the new and old commands still need to use the current register in the same pipeline stage. In such case, a field branch is created, i.e. the field of the new command is inserted into the pipeline, while the field of the old command enters into the field branch, and such branch shall maintain until the last time the old command uses this field for operation. Fig. 7 illustrates the hardware structure of the field branch of one pipeline stage, and the other stages are same as this one. The so called field branch means there are two routes of current registers, one act as the major current register 71, while the other act as the field branch register 72, which are connected to the input of the processing module 74 through a multi-route switch.

As discussed above, the commands that could be overlapped will not incur conflict of processing module. In case new command shall be processed, the processing module will receive the input of the major current register, while in case the old command shall be processed, the processing module will receive the input from the field branch register, and the register not necessary to be processed can directly enter the similar current register in the next pipeline; after the last time the old command uses this field, the value of the field branch register is then deserted. The judgment on the field conflict is made at the same time of the judgment of relevance, that is, at the stage before inserting the new command, and the old command enters into the field branch at the first stage.

Referring to both Fig. 1 and Fig. 5, in the overlapping command committing method according to the present invention, the Command G is inserted in the last cycle of the Command A before exiting the pipeline, avoiding the appearance of bubble, which can save up to six clock periods for inserting the following command into the pipeline compared with the conventional command committing method of inserting processor, and the waiting interval for committing commands is also reduced by 6 clock periods. Therefore, the new command inserting method in the present invention – the overlapping command committing method can effectively improve the utilization efficiency of every element of the pipeline and further increase the speed of the command execution.

INDUSTRIAL APPLICABILITY

Although the above embodiment is an example of a particular kind of communications chip, it is obvious that the overlapping command committing method of dynamic cycle pipeline is not limited to one or several kinds of command(s), and thus can be applied to other chips with the cycle pipeline structure wherein the entering stage and exiting stage of the pipeline commands are not the same stage.